

# CPE 325: Embedded Systems Laboratory

## Laboratory Tutorial #1:

### Introduction to TI Code Composer Studio (IDE)

Aleksandar Milenković

Email: [milenska@uah.edu](mailto:milenska@uah.edu)

Web: <http://www.ece.uah.edu/~milenska>

#### Table of Contents

1 Creating a Project.....	1
1.1 Creating a New Workspace and a Project .....	1
1.2 Compiling and Linking the Application .....	4

## Objectives

This tutorial will help you get started with the TI's Code Composer Studio for MSP430. It includes the following topics:

*Creating an application project*

### Notes:

The latest version of Code Composer Studio can be downloaded for free from the TI's web site: <http://www.ti.com/tool/ccstudio>.

## Prerequisites

Active knowledge of C/C++ programming language and understanding of computer architecture.

## Expected Outcomes

Upon completion of this tutorial you will be able to create your own projects using TI's Code Composer Studio. You will understand main steps of software development for embedded systems. Specifically, you will be able to:

- Create a workspace and a project using TI's Code Composer Studio
- Compile source files, download the executable, and run the program on an embedded platform

## 1 Creating a Project

This section introduces you to TI's Code Composer Studio Integrated Development Environment (IDE) that will be used for software development in the Embedded Systems Laboratory. In the form of a step-by-step tutorial, it demonstrates a typical development cycle and shows you how to use the CCStudio compiler and the CCStudio linker to create a small application for the MSP430 microcontroller. It includes topics such as creating a workspace, setting up a project with C source files, compiling and linking your application, and debugging.

### 1.1 Creating a New Workspace and a Project

Using the Code Composer Studio IDE, you can design advanced project models. For more information, read the Code Composer Studio wiki. <http://processors.wiki.ti.com/index.php/Category:CCS>

Here, we will walk through a relatively simple project with several source files.

Step 1. Open Code Composer Studio (CCS).

Step 2. In the Eclipse Launcher select a directory to be used as the workspace (e.g., Figure 1.). press Launch. A Getting Started window will appear (Figure 2). This page provides links to example projects as well as documentation related to Code Composer and TI Microcontrollers.

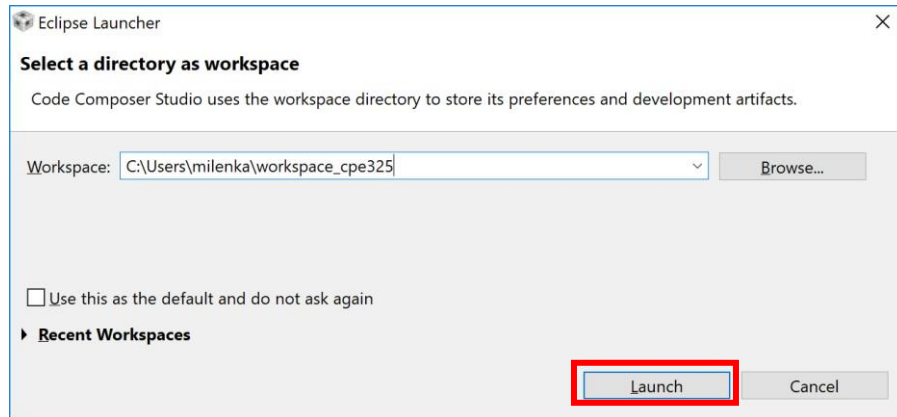


Figure 1. Eclipse Launcher

Step 3. Create a new project from the Getting Started menu. Click the *New Project* button. A New CCS Project window appears as shown in Figure 3.

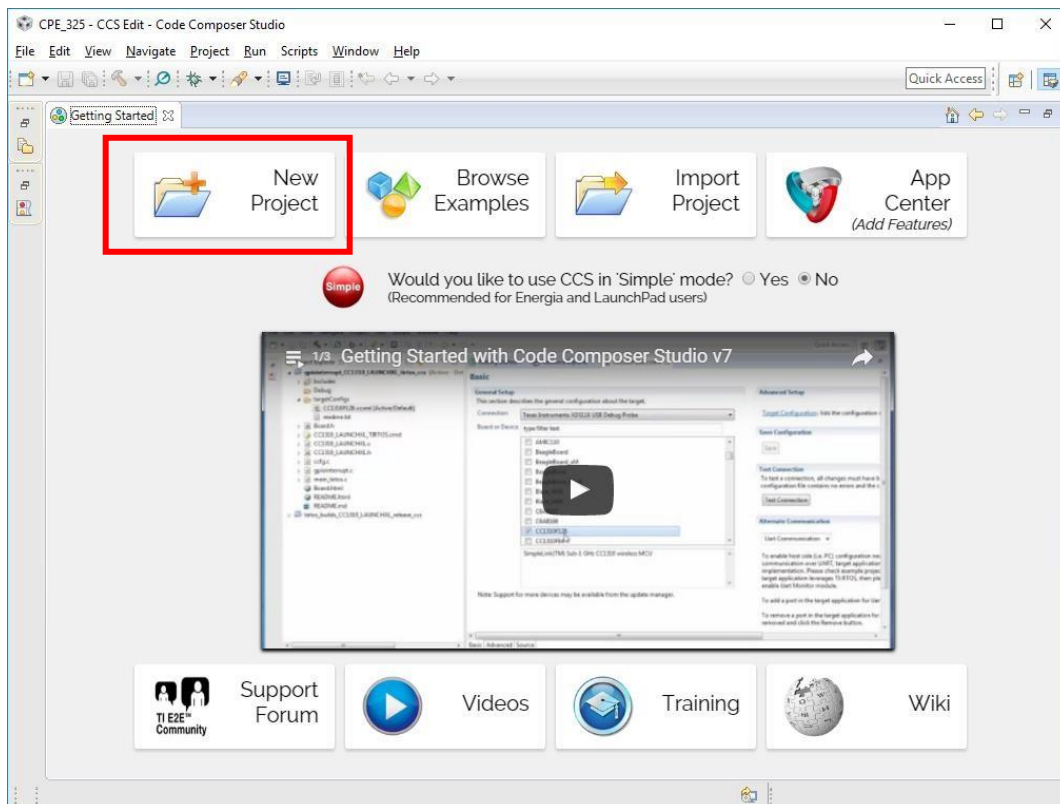


Figure 2. CCS Getting Started Page

Step 4. Name the project "Lab1\_D1" and select "Empty Project" (see Figure 3.). For the target select one of the following options depending on the target board: (a) MPS430FG4618 for the MSP430 Experimenter board; or MSP430G2553 for the MSP430 Launchpad.

Step 5. Click <Finish>. The Project Explorer will show where all the project files are, note there are no source code files because we created an Empty Project (Figure 4).

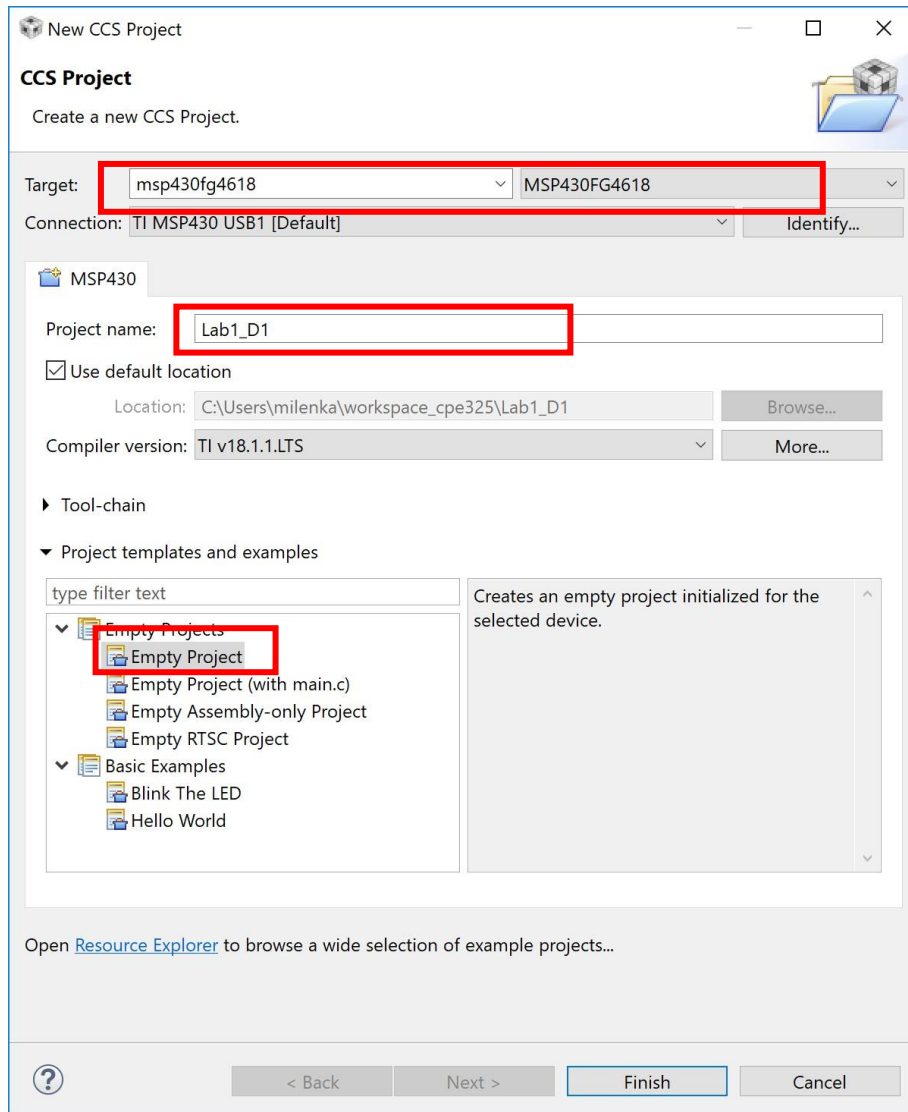


Figure 3. New CCS Project Setup

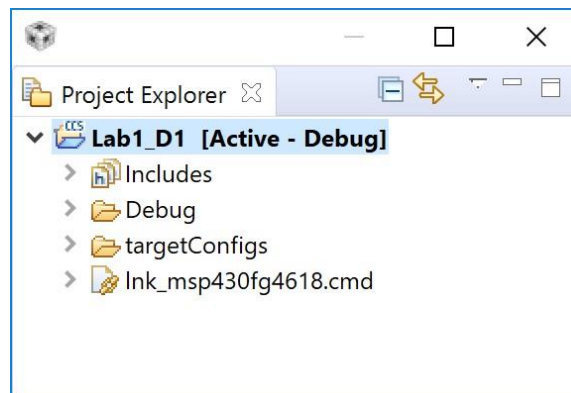


Figure 4 Project Explorer

Step 6. Copy the “Lab1\_D1.c” and “twofact.c” from Windows Explorer to Lab\_D1 in the project explorer. Right click on Lab1\_D1 and select Add Files option. When prompted select “Copy Files” and click <OK>.

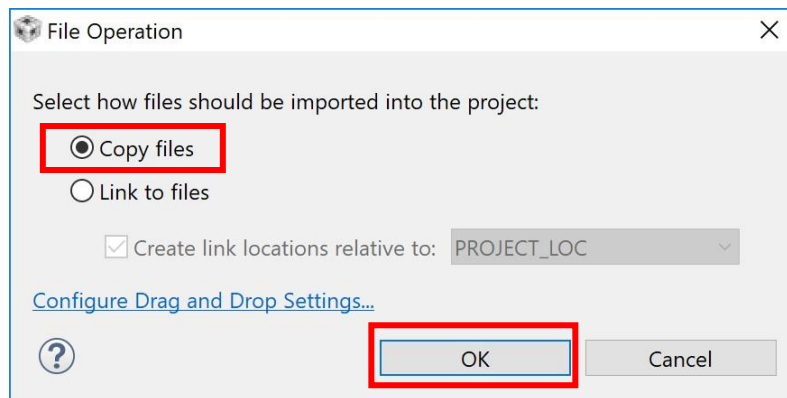


Figure 5. File Operation Dialog

Step 7. Double-click one of the files to see the source code (see Figure 6).

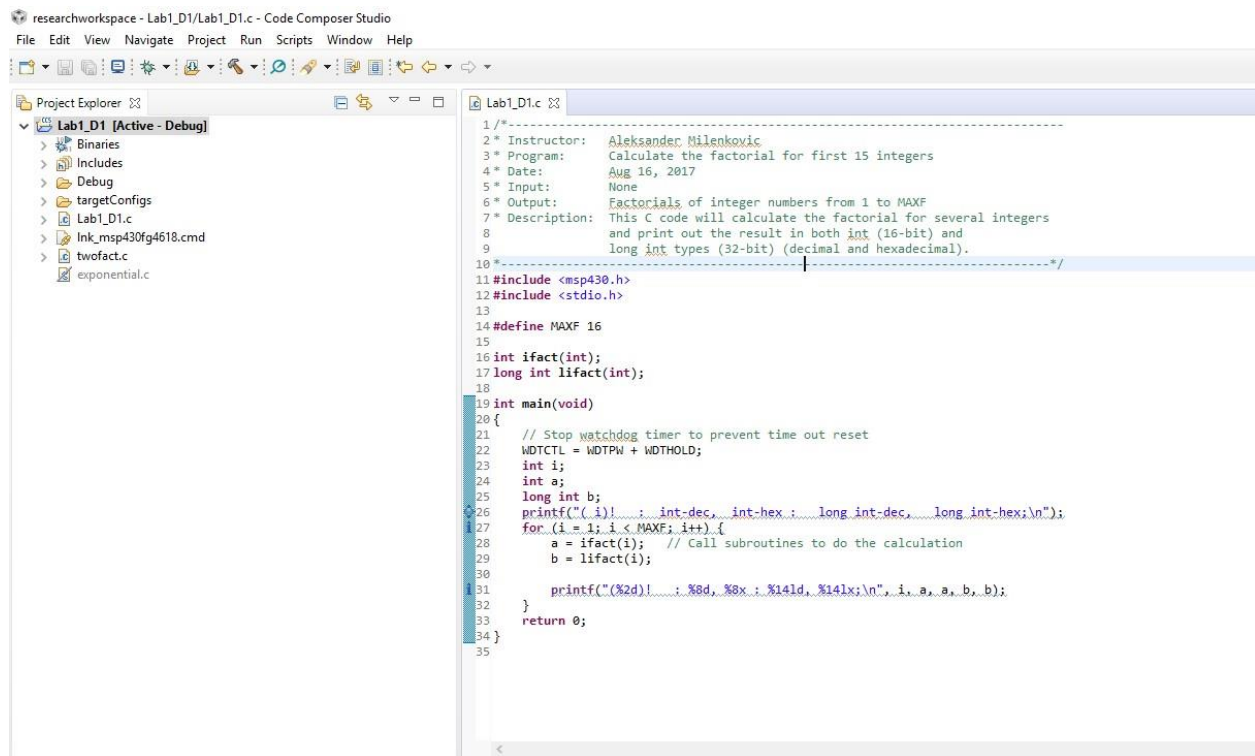


Figure 6. Code Composer Studio.

## 1.2 Compiling and Linking the Application

You can now compile and link the application. You should also create a compiler list file and a linker map file and view both.

Step 1. Right-Click “Lab\_D1” in the project explorer and select properties. A new window will pop up as shown in Figure 7.

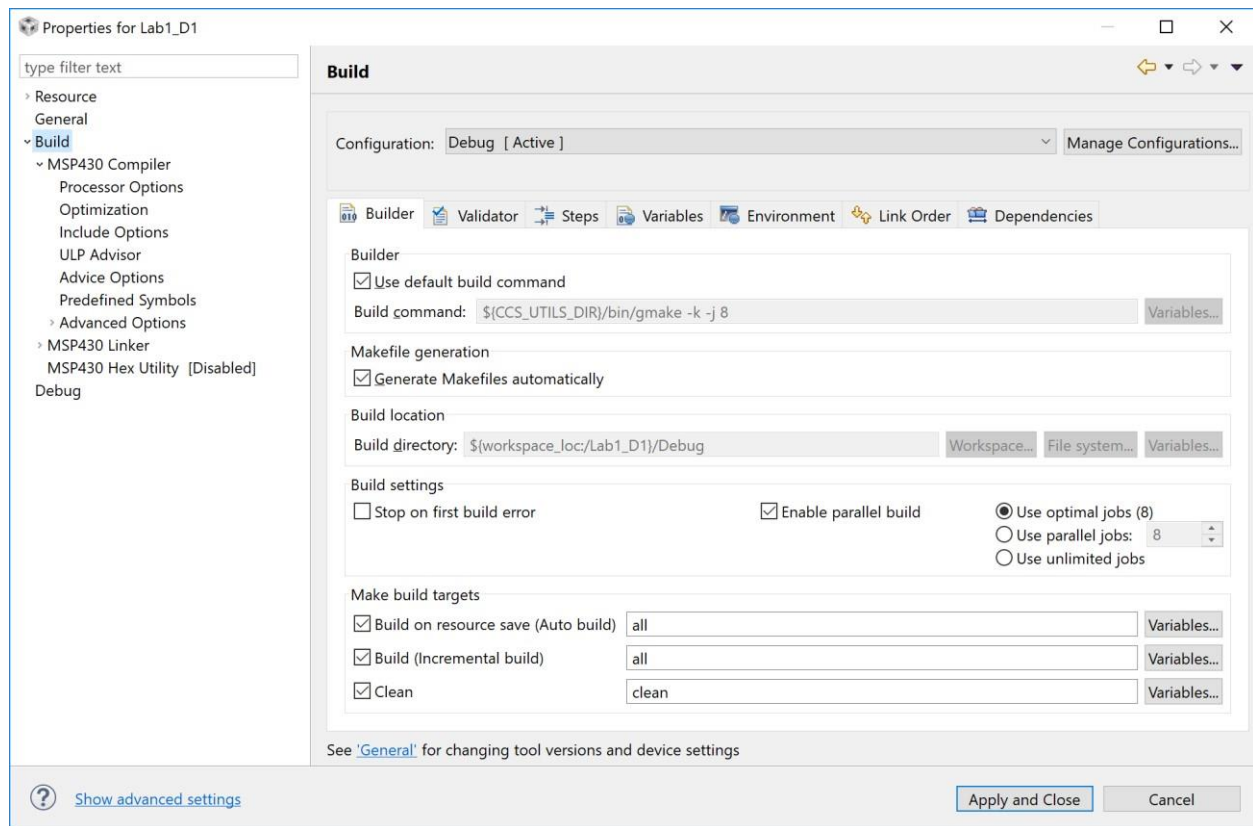


Figure 7. Project Properties

Step 2. From this menu, you can setup various project settings such as optimizations, list file generation, heap size, stack size, and the level of printf/scanf support. Below are several useful options.

- a. Select the “Build->MSP430 Compiler->Optimization.” This option allows you to set optimization settings. Change the “Optimization level” to off.
- b. Select the “Build->MSP430 Compiler->Advanced Options->Assembler Options” to setup the listing file options. Click the checkbox “Generate listing file (--asm\_listing, -al)” (Figure 8).
- c. Select the “Build->MSP430 Compiler->Processor Options.” This allows you to set the code and memory model. Set the “Silicon version” to msp and the “code memory model” and “data memory model” to small. This will limit the instruction set to non-extended instructions.

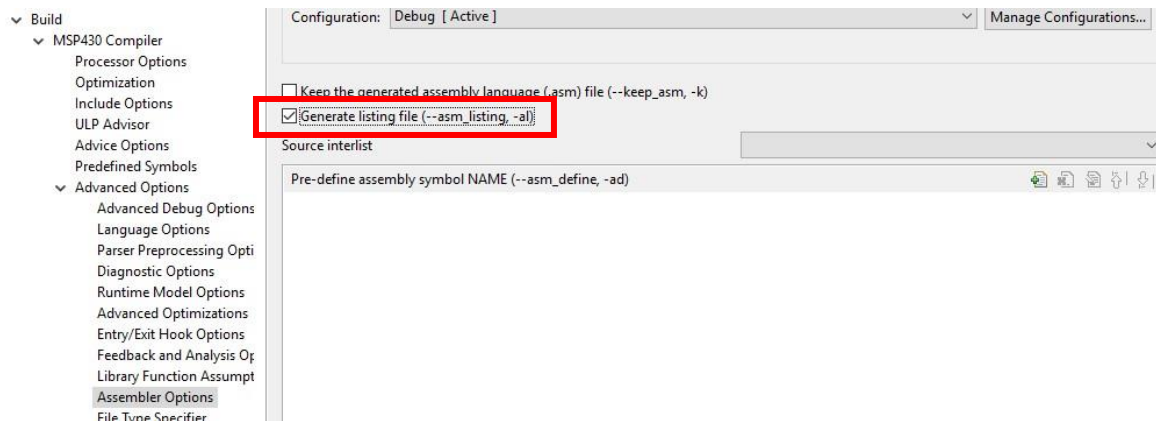


Figure 8 Project Assembler Options

- d. Select the “Build->MSP430 Compiler->Advanced Options->Language Options”. Change “Level of printf/scanf support requires (--printf\_support) to full.

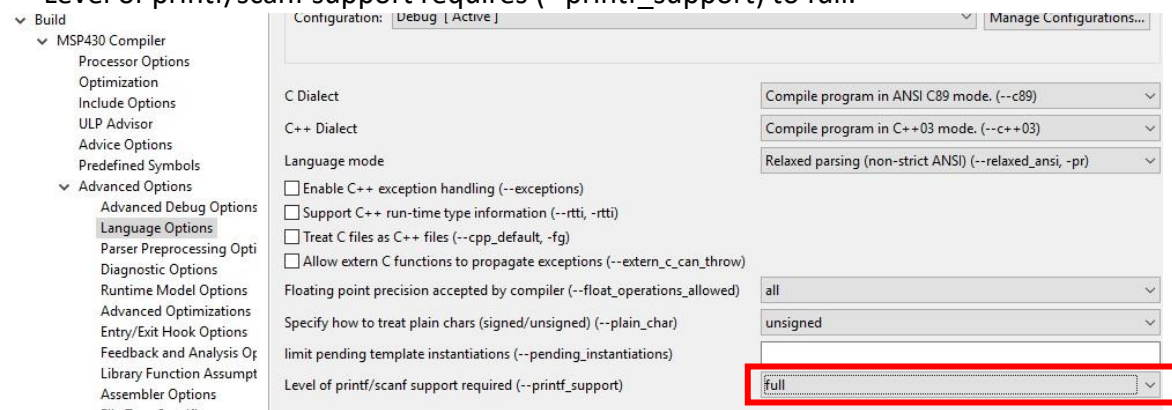


Figure 9. Compiler Language Options

- e. Select the “Build->MSP430 Linker->Basic Options” to set the heap size. Change “Heap size for C/C++ dynamic memory allocation (--heap\_size, -heap) from 80 to 300. This change is necessary to get the printf debugging to work later in the lab. Click <Apply and Close> to close the properties windows.

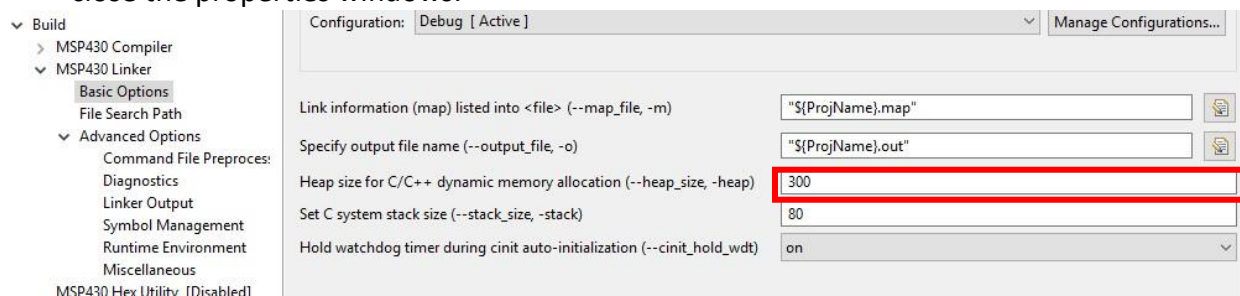


Figure 10 Project Linker Options

Step 3. To build a single file right-click the file in the Project Explorer and select “Build Selected File(s)” as shown in Figure 11. The Console window will show you if there are any build errors (Figure 12).



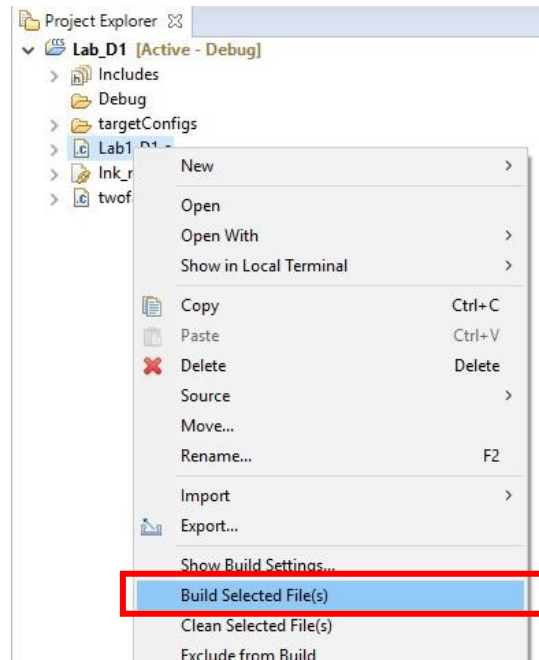


Figure 11. File Options Menu

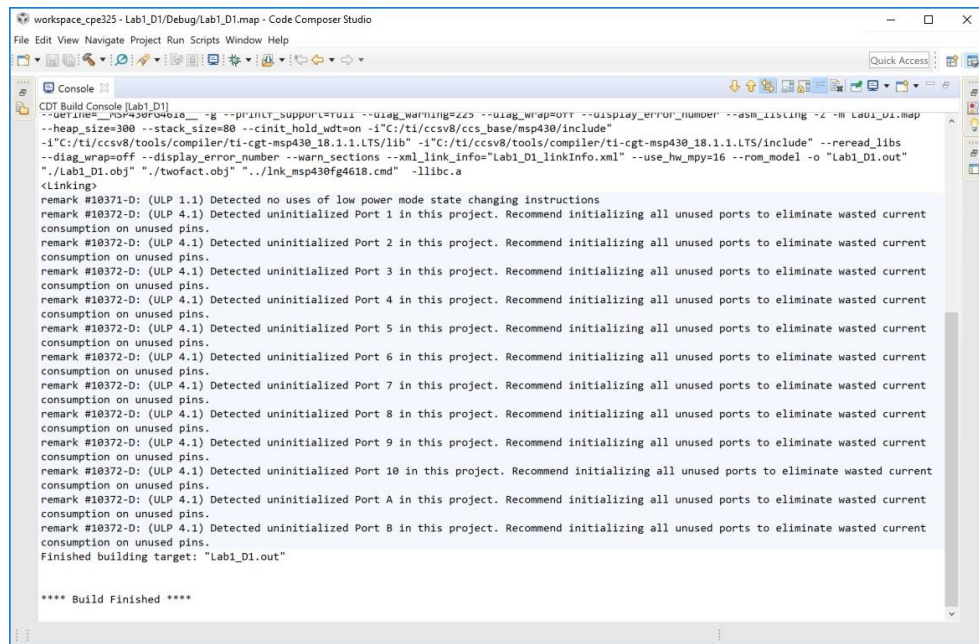



Figure 12 CDT Build Console

Step 4. Repeat the steps to build the “twofact.c” file. Click the  icon to build and link the project.